A CRM Application To Handle The Clients And Their Property Related Requirements

Personal Information

- **Name:** Chandaka Koushik
- **Email:** 323103384l16@gvpce.ac.in
- **College Name:** Gayatri Vidya Parishad College of Engineering (Autonomous)

Project Overview

Dreams World Properties integrates Salesforce to streamline customer interactions. The website engagement triggers automated record creation in Salesforce, capturing customer details and preferences. Users are categorized as approved or non-approved, providing tailored property selections to approved users. This integration optimizes operations, enhances user experience, and facilitates growth in the real estate market.

Objectives

Business Goals:

- **Improve Client Management**: Maintain a comprehensive record of client information, preferences, and interactions.
- **Enhance Customer Satisfaction**: Provide personalized service by quickly addressing client property requirements.
- **Streamline Property Management**: Efficiently handle property listings, viewings, and transactions.
- **Boost Sales Efficiency**: Enable sales teams to track and manage leads and opportunities more effectively.
- **Data-Driven Insights**: Generate reports and insights to understand market trends and client needs.

Specific Outcomes:

- **Faster Client Response Times**: Reduced response times to client inquiries.
- **Increased Conversion Rates**: More effective lead tracking and follow-ups.
- **Improved Customer Retention**: Enhanced relationships with personalized interactions.
- **Increased Productivity**: Sales and service teams spend less time on manual data entry and more on client engagement.
- **Better Market Understanding**: Enhanced analytics for market trends and demand prediction.

Salesforce Key Features and Concepts Utilized:

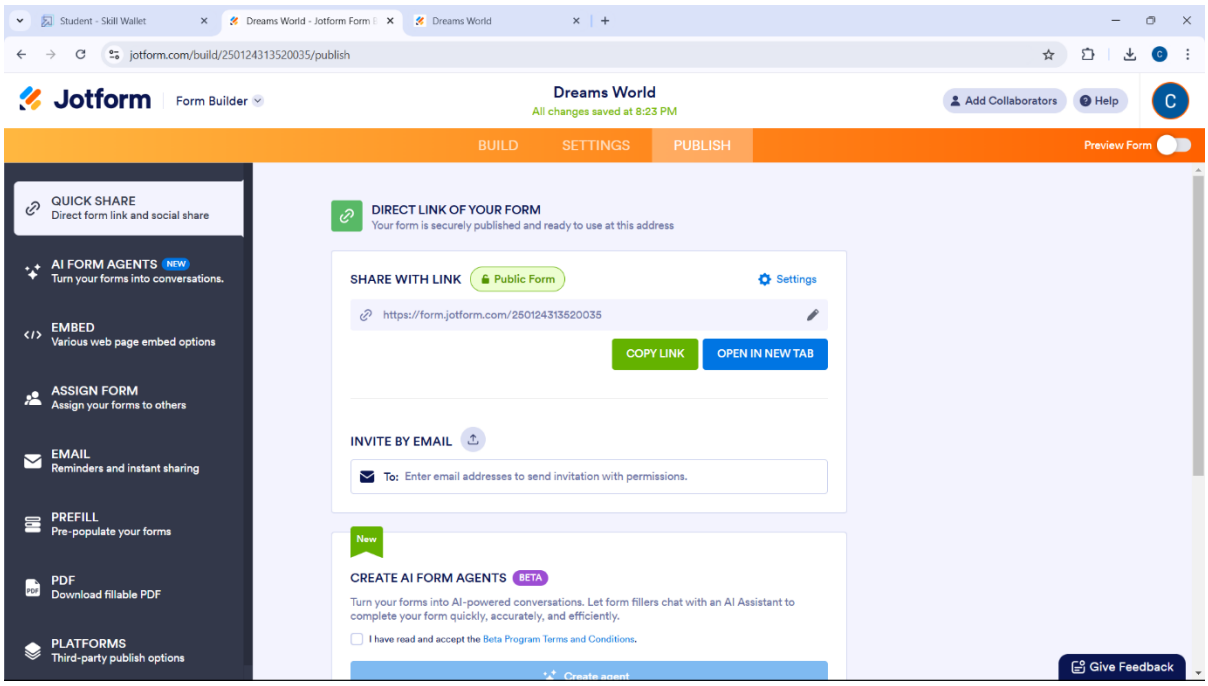
- **Sales Cloud**: Manage leads, opportunities, and accounts.
- **Service Cloud**: Track customer inquiries and provide support.
- **Property Management Objects**: Custom objects for properties, listings, and client requirements.
- **Reports and Dashboards**: Visualize key metrics for property and client performance.
- **Mobile Accessibility**: Manage CRM from any device for on-the-go updates.
- **Automation and Workflow Rules**: Automate notifications and follow-up reminders.

Solution Design

Step 1: Create a Jotform and Integrate with Salesforce

1. Login to Jotform and create a form with fields like Name, Phone, Email, Address, and Property Type.
2. Publish the form and integrate it with Salesforce to create customer records.

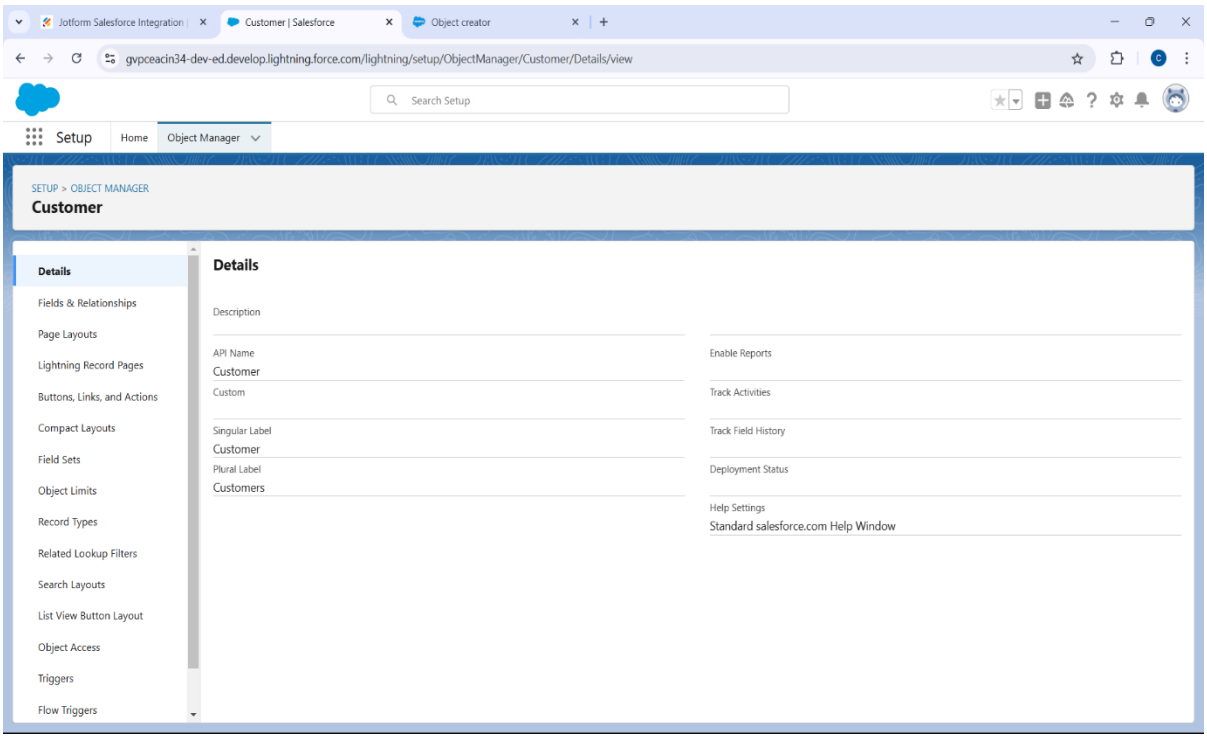
https://form.jotform.com/250142119167046



Step 2: Create Objects from Spreadsheets

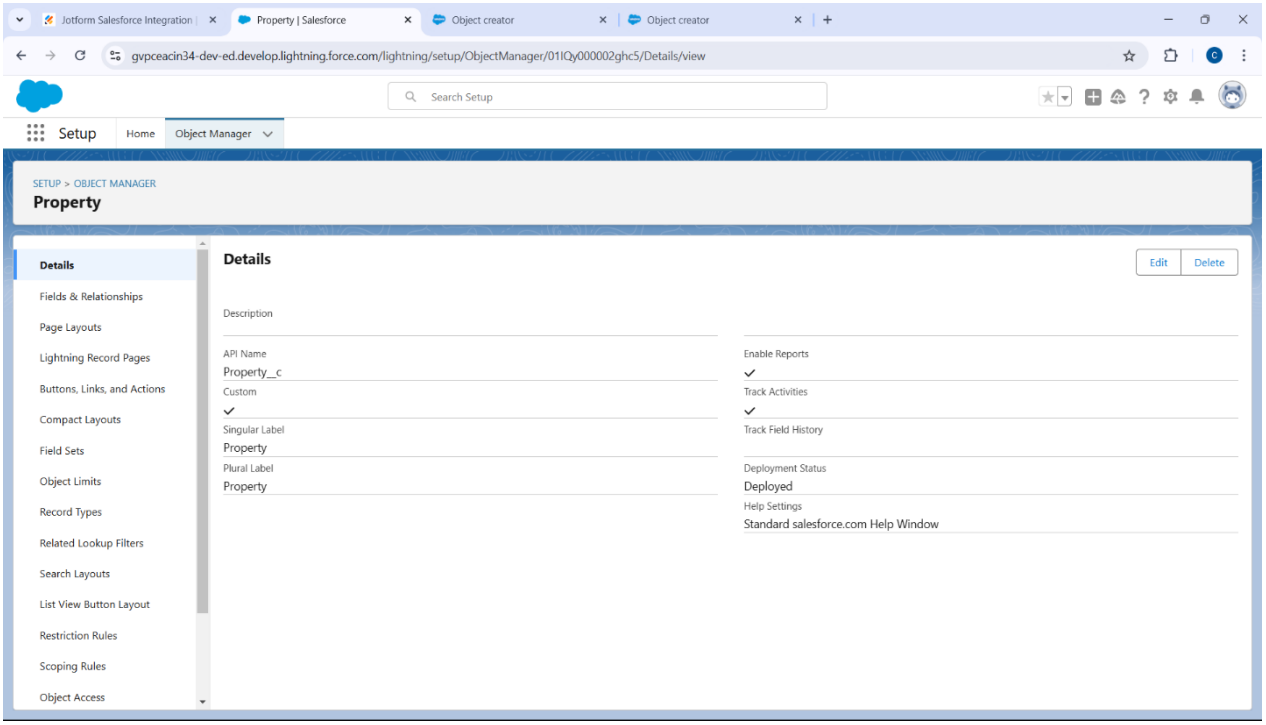
Customer Object:

- Upload a spreadsheet with customer details and map the fields to create the object.



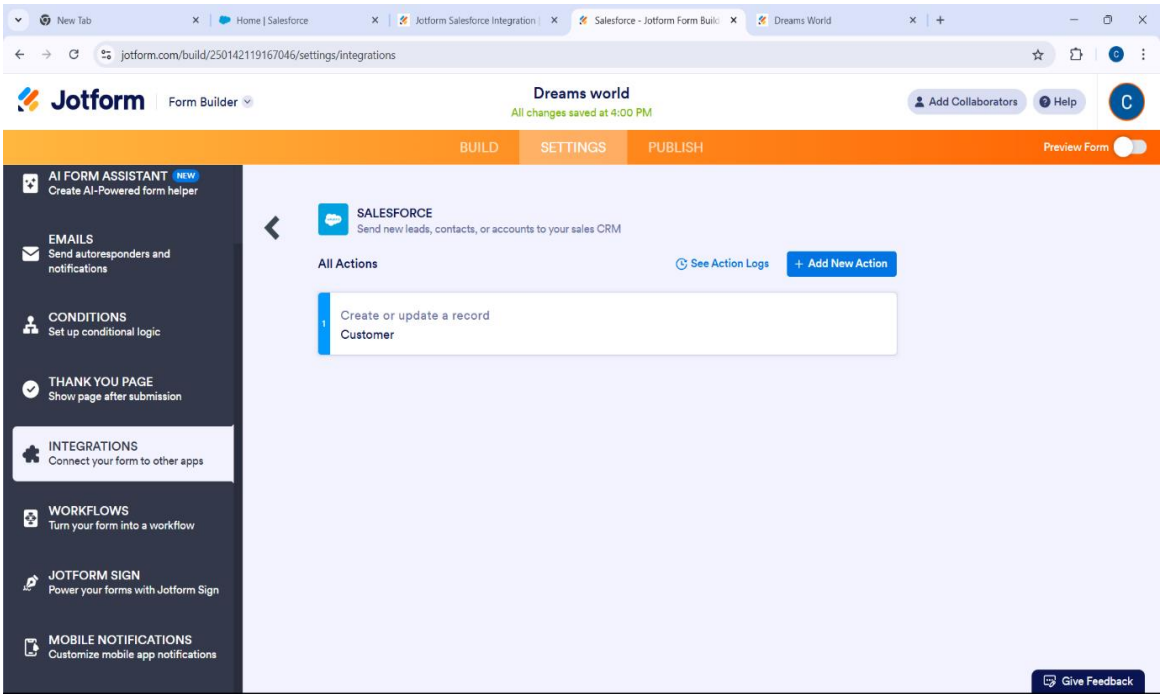
Property Object:

- Repeat the process for property data.



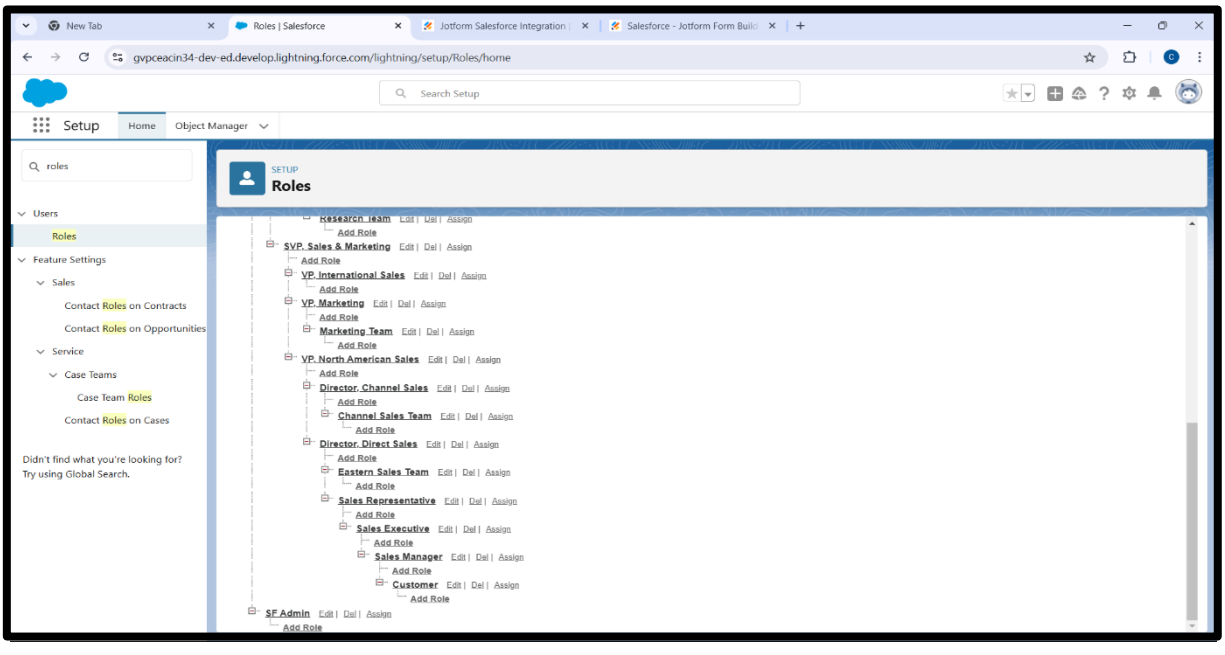
Step 3: Integrate Jotform with Salesforce

1. On Jotform, choose Salesforce Integration.
2. Select the desired Salesforce org and action (e.g., create a record).
3. Map form fields to Salesforce object fields.
4. Save the integration.



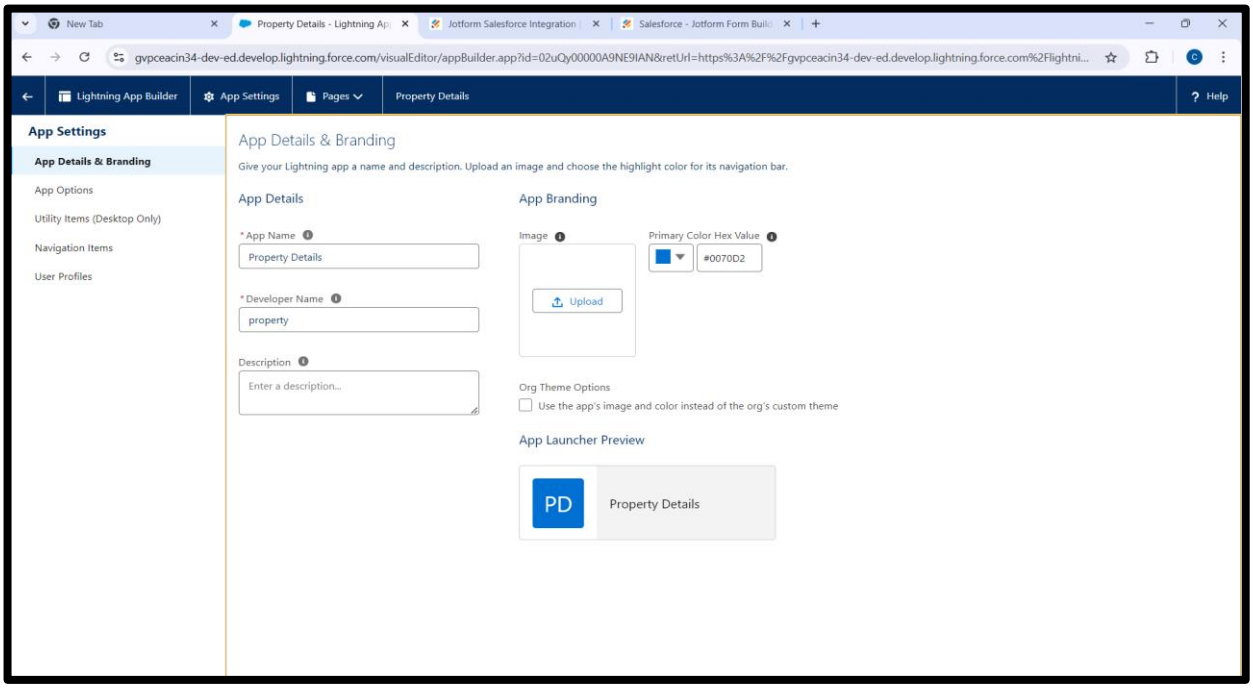
Step 4: Define Roles

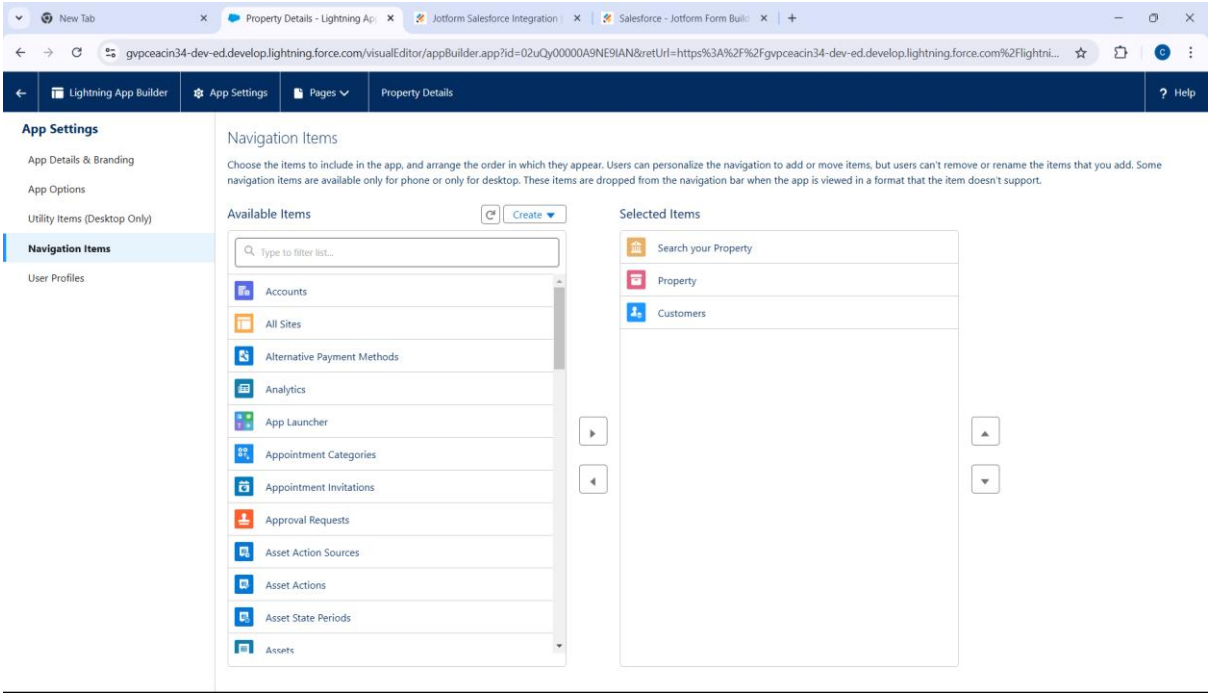
- **Sales Executive:** Reports to Sales Representative.
- **Sales Manager:** Reports to Sales Executive.
- **Customer:** Reports to Sales Manager.

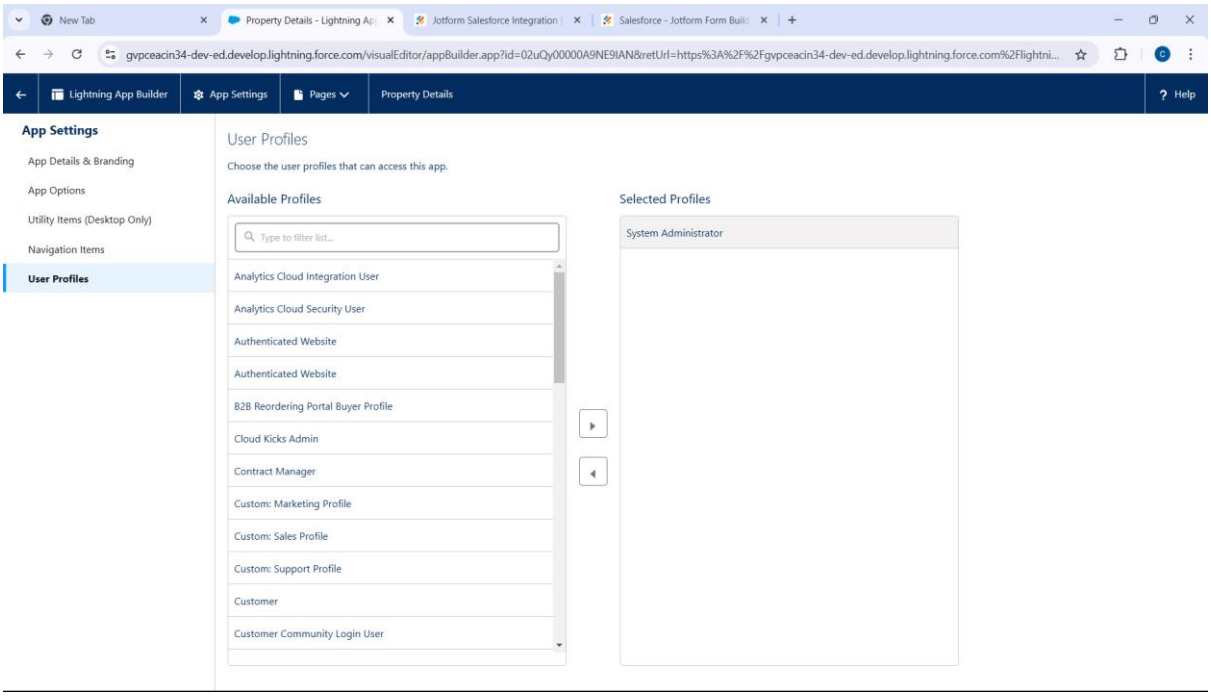


Step 5: Create a Property Details App

1. Go to App Manager in Salesforce Setup and create a Lightning App.
2. Include the "Customer" and "Property" objects.
3. Assign the app to the System Admin profile.



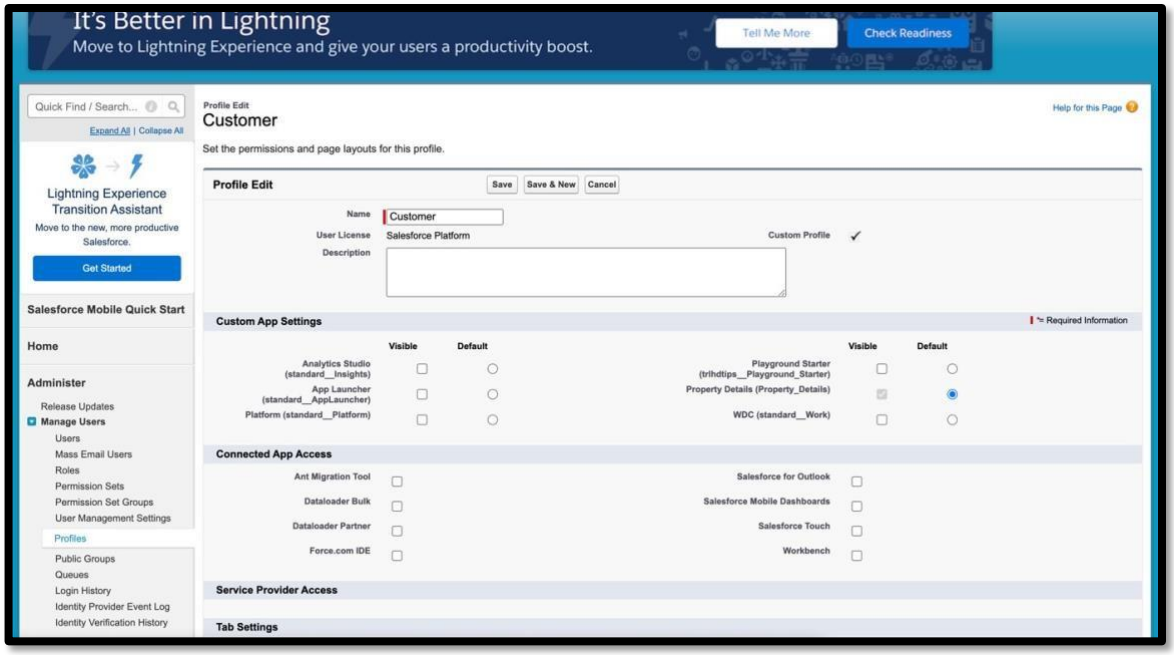




Step 6: Create Profiles

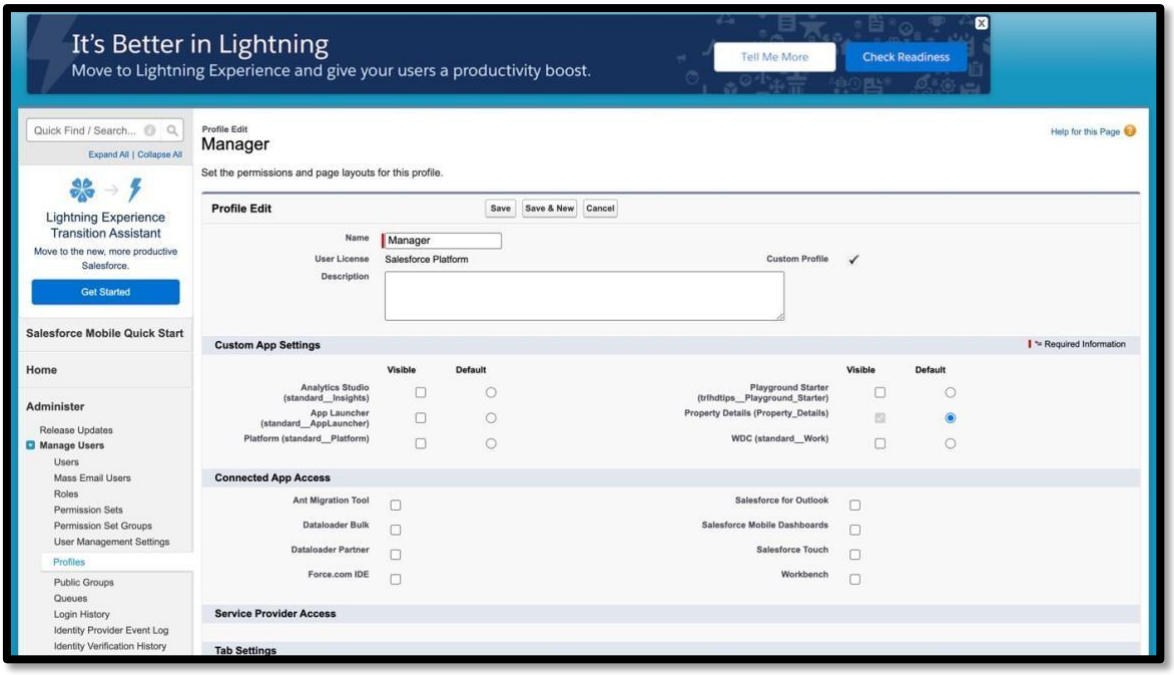
Customer Profile:

- Clone the Salesforce Platform User profile.
- Restrict access to custom objects except for "Property Details."
- Allow read and view-all permissions on "Property."



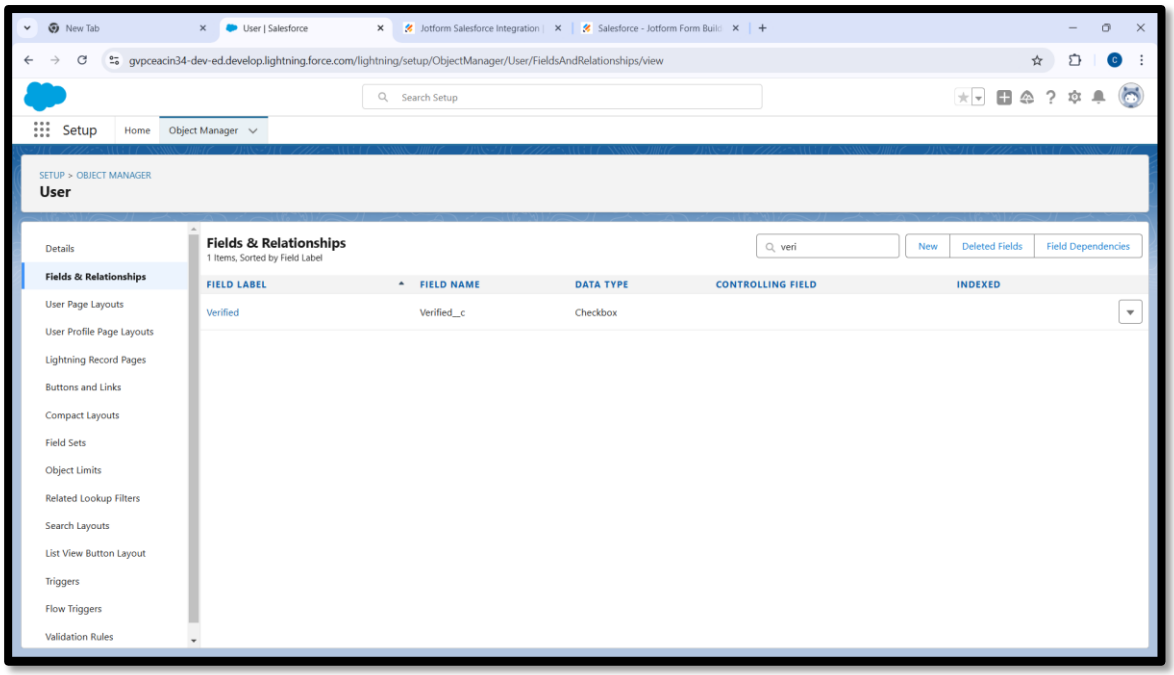
Manager Profile:

- Clone the Salesforce Platform User profile.
- Restrict access except for "Property Details."
- Grant "Modify All" permissions on "Property" and "Customer."



Step 7: Add a Checkbox Field

1. Navigate to the User Object Manager in Setup.
2. Create a new field named "Verified" with data type "Checkbox."



Step 8: User Creation

User 1

- **Name:** Executive
- **Role:** Sales Executive
- **License:** Salesforce Platform
- **Profile:** System Administrator

User 2

- **Name:** Manager
- **Role:** Sales Manager
- **License:** Salesforce Platform
- **Profile:** Manager

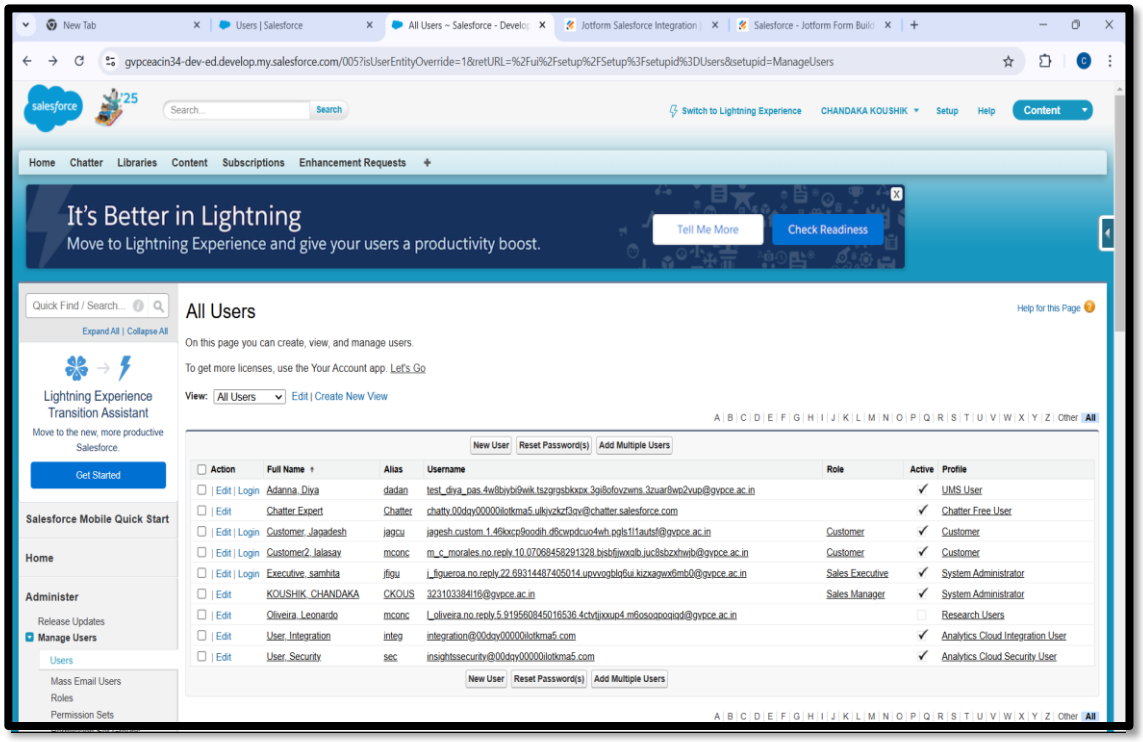
User 3

- **Name:** Customer
- **Role:** Customer
- **License:** Salesforce Platform
- **Profile:** Customer
- **Verified Checkbox:** Unchecked

User 4

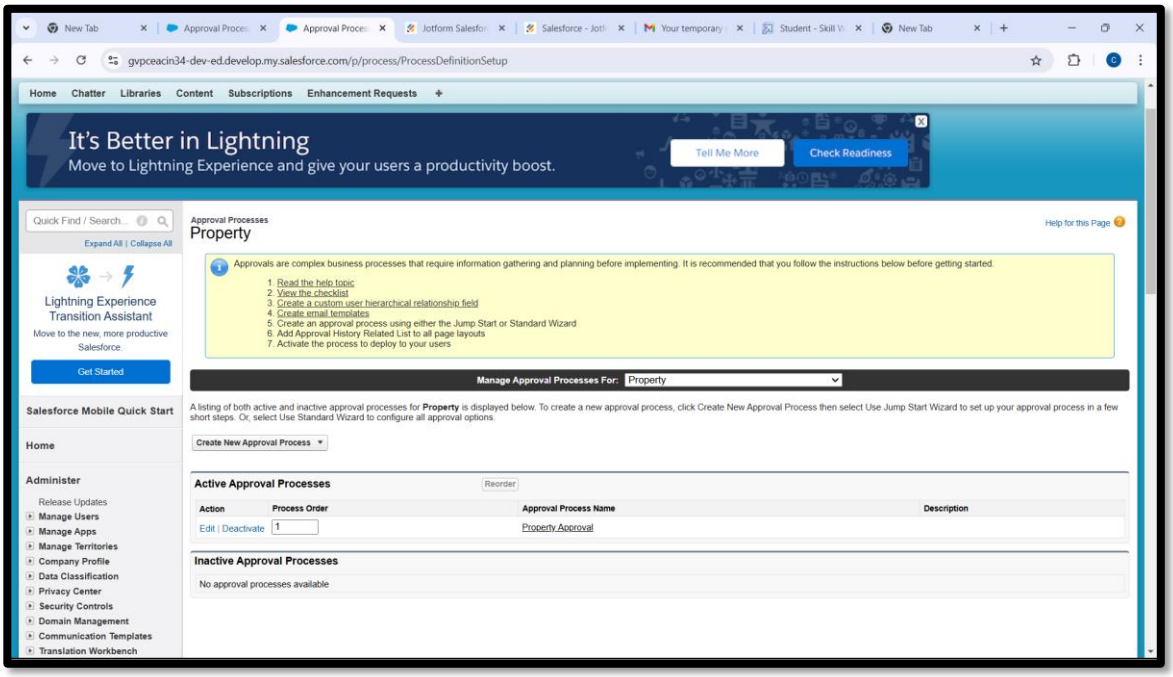
- **Name:** Customer2
- **Role:** Customer
- **License:** Salesforce Platform
- **Profile:** Customer
- **Verified Checkbox:** Checked

Save



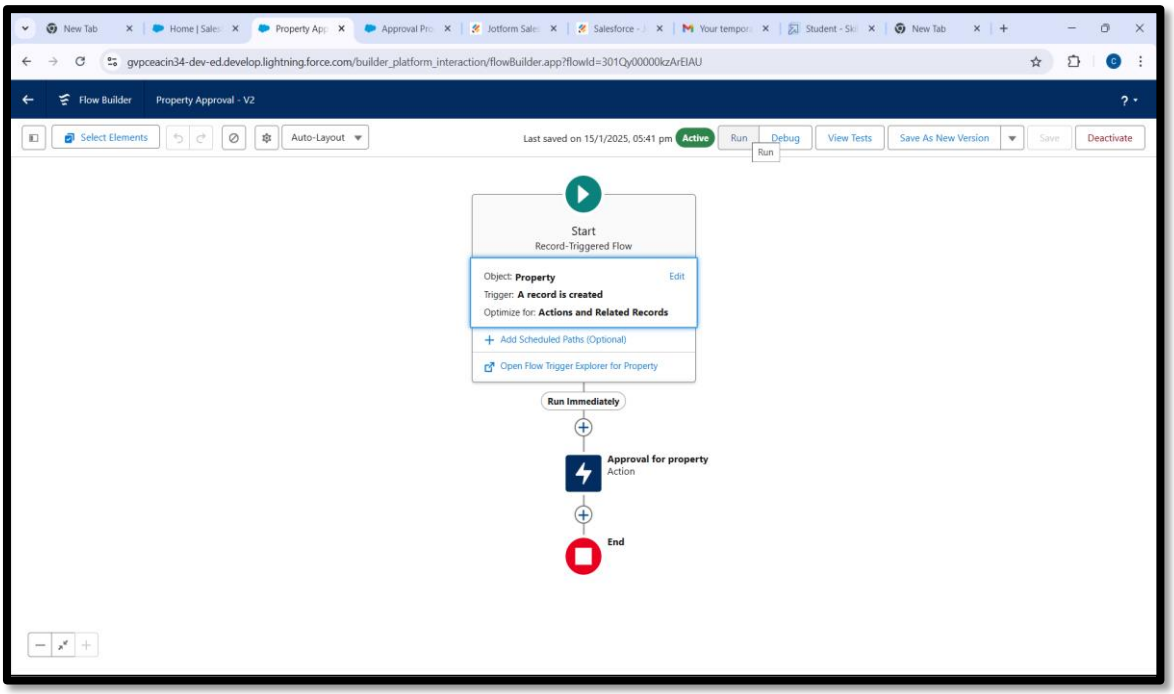
Step 9: Approval Process for Property Object

1. **Process Name:** Property Approval
2. **Criteria:**
    - Location is not blank.
    - Verified equals false.
3. **Approver:** Manager
4. **Approval Steps:**
    - Step 1: Executive Approval (Sales Executive as approver).
5. **Field Updates:**
    - Verified Property: Set checkbox to true.
    - Unverified Property: Set checkbox to false.
6. Activate the approval process.



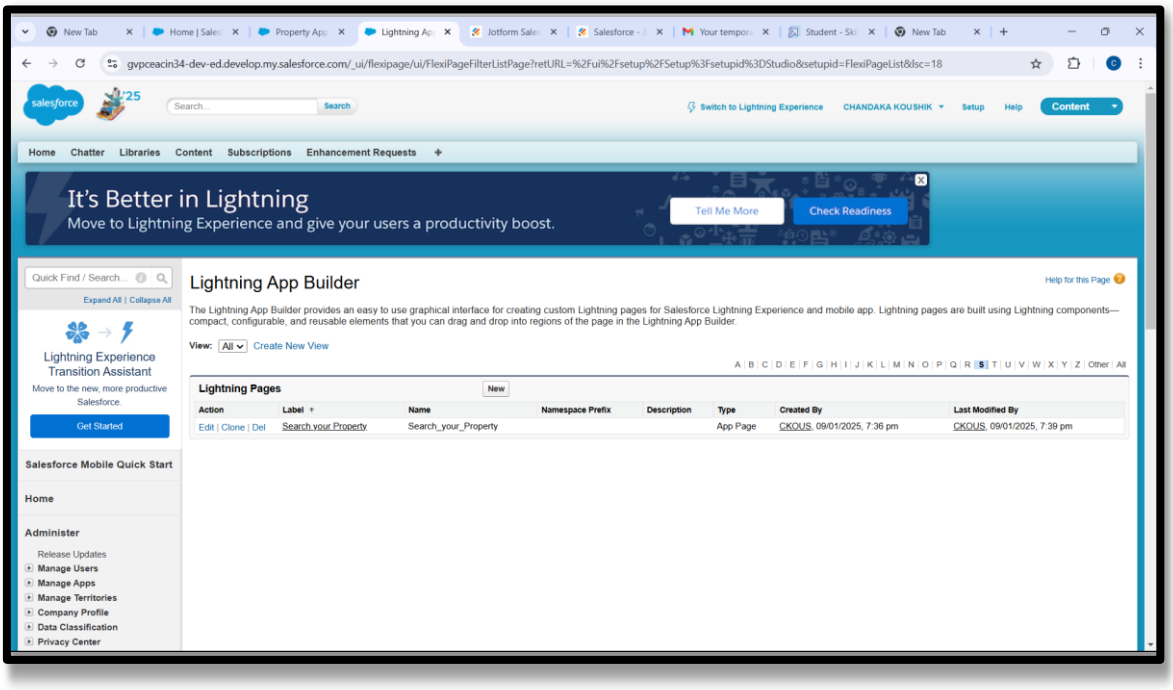
Step 10: Record Trigger Flow

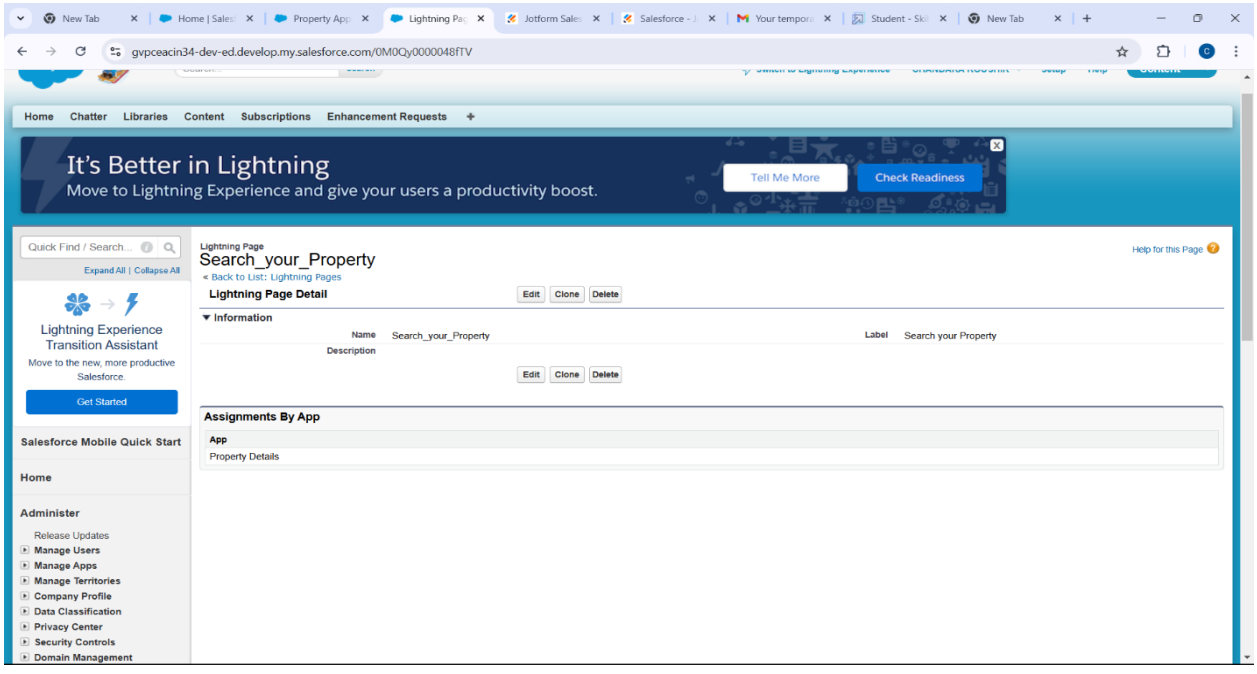
1. Create a Record Trigger Flow.
2. Set trigger conditions to "Record Created."
3. Add an action to "Submit for Approval."
4. Save and activate the flow



Step 11: Create an AppPage

- From Setup >> Go to Lightning App Builder >> Click on New >> Select App Page and Click on Next.
- Give Label as “Search your Property” click “Next”.
- Click “header and Left Side bar” and Click on “Done”.
- Click on “Save” and then click on “Activate”.
- From Page Setting select page activation as “Activate for all Users”.
- From Lightning Experience Click on “Property Details” and click on Add Page.
- Then Click on “Save”.

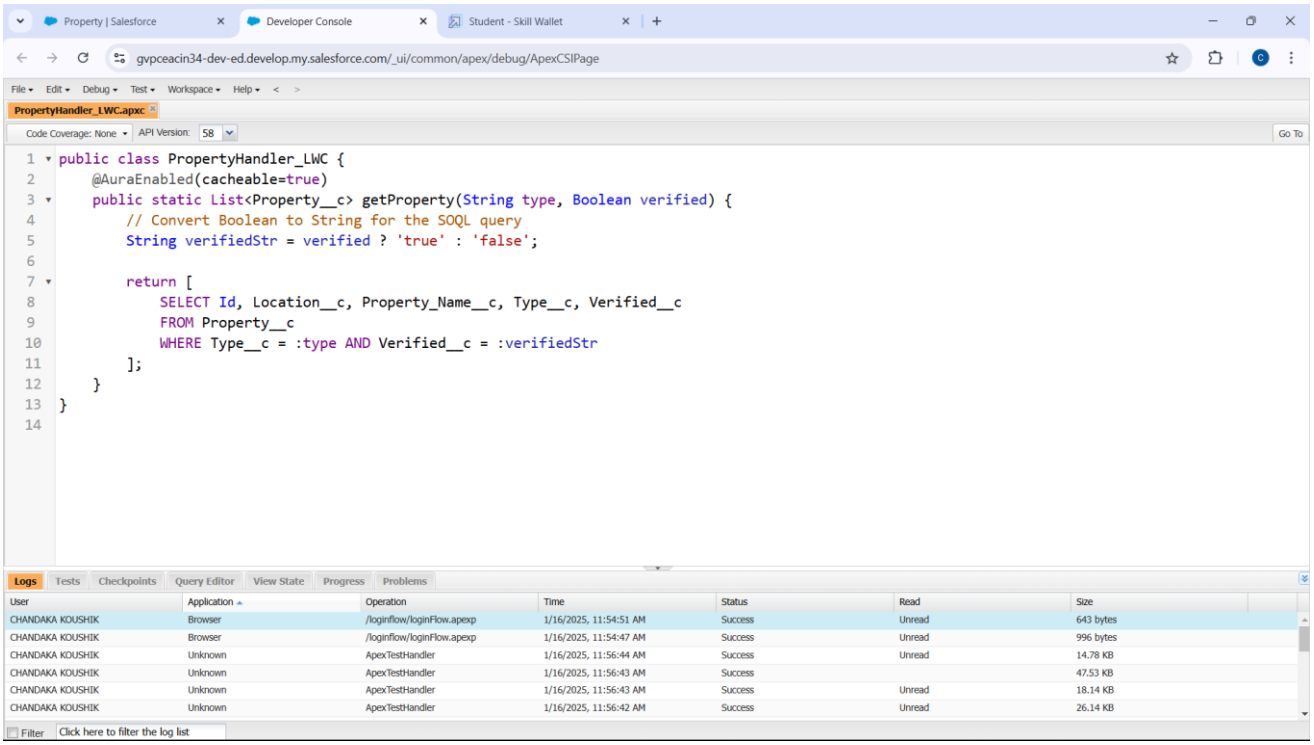




Step 12: Lightning Web Component (LWC)

Apex Class

```
public class PropertyHandler_LWC {
    @AuraEnabled(cacheable=true)
    public static List<Property__c>getProperty(String type, Boolean verified) {
        String verifiedStr = verified ? 'true' : 'false';
        return [
            SELECT Id, Location__c, Property_Name__c, Type__c, Verified__c
            FROM Property__c
            WHERE Type__c= :type AND Verified__c = :verifiedStr
        ];
    }
}
```



- Create a Lightning Web Component in your Vs Code and (ctrl+shift+P) and click on authorize an org.
- Enter your login id and password to authorize your org.
- Now (ctrl+shift+P) and Create a lightning Web Component and Name it Anything you want to. (Example -)

LWC Code

HTML

```
<template>
   <lightning-card>
      <div class="slds-box">
         <div class="slds-text-align_left">
            <h1 style="font-size: 20px;"><b>Properties</b></h1>
         </div>
         <div>
            <div class="slds-grid slds-gutters">
               <div class="slds-col slds-size_5-of-6">
                  <lightning-combobox
                     name="Type"
                     placeholder="Select Property type"
                     value={typevar}
                     options={propertyoptions}
                     onchange={changehandler}>
                  </lightning-combobox>
               </div>
               <div class="slds-col slds-size_1-of-6">
                  <br>
                  <lightning-button-icon
                     icon-name="standard:search"
                     variant="neutral"
                     alternative-text="Search"
                     label="Search"
                     onclick={handleClick}>
                  </lightning-button-icon>
               </div>
            </div>
         </div>
      </div>
```

```
        <!-- Display property list or 'No properties found' message -->
        <template if:true={istrue}>
            <div class="slds-box">
                <lightning-datatable
                    key-field="id"
                    data={propertylist}
                    columns={columns}>
                </lightning-datatable>
            </div>
        </template>

        <template if:false={isfalse}>
            <div class="slds-box">
                <div style="font-size: 15px;"><b>No properties are found!!</b></div>
            </div>
        </template>
    </lightning-card>
</template>
```

JavaScript

```
import { LightningElement, api, track, wire } from 'lwc';
import getProperty from '@salesforce/apex/PropertyHandler_LWC.getProperty'; // Corrected
import { getRecord } from 'lightning/uiRecordApi';
import USER_ID from '@salesforce/user/Id';

export default class PropertyManagement extends LightningElement {
    @api recordId;
    userId = USER_ID;
    @track verifiedvar = false;
    @track isfalse = true;
    @track istrue = false;
    @track typevar = '';
    @track propertylist = [];
```

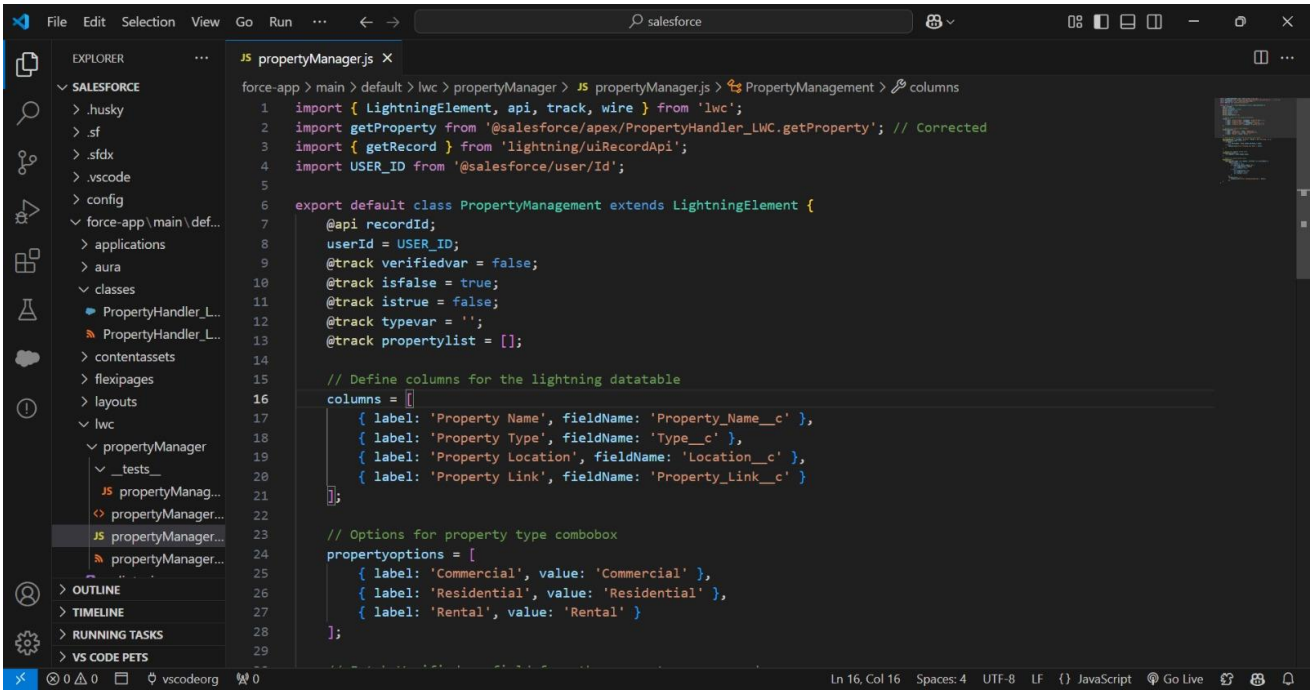
```
// Define columns for the lightning datatable
columns = [
    { label: 'Property Name', fieldName: 'Property_Name__c' },
    { label: 'Property Type', fieldName: 'Type__c' },
    { label: 'Property Location', fieldName: 'Location__c' },
    { label: 'Property Link', fieldName: 'Property_Link__c' }
];

// Options for property type combobox
propertyoptions = [
    { label: 'Commercial', value: 'Commercial' },
    { label: 'Residential', value: 'Residential' },
    { label: 'Rental', value: 'Rental' }
];

// Fetch Verified__c field from the current user record
@wire(getRecord, { recordId: "$userId", fields: ['User.Verified__c'] })
recordFunction({ data, error }) {
    if (data) {
        this.verifiedvar = data.fields.Verified__c.value;
    } else {
        console.error('Error fetching user data:', error);
    }
}

// Handler for combo box change event
changehandler(event) {
    this.typevar = event.target.value;
}

// Handler for search button click
handleClick() {
```

```
        getProperty({ type: this.typevar, verified: this.verifiedvar })
            .then((result) => {
                this.istrue = true;
                if (result && result.length > 0) {
                    this.propertylist = result;
                    this.isfalse = true;
                } else {
                    this.propertylist = [];
                    this.isfalse = false;
                    this.istrue = false;
                }
            })
            .catch((error) => {
                console.error('Error fetching properties:', error);
            });
    }
}
```

Metadata

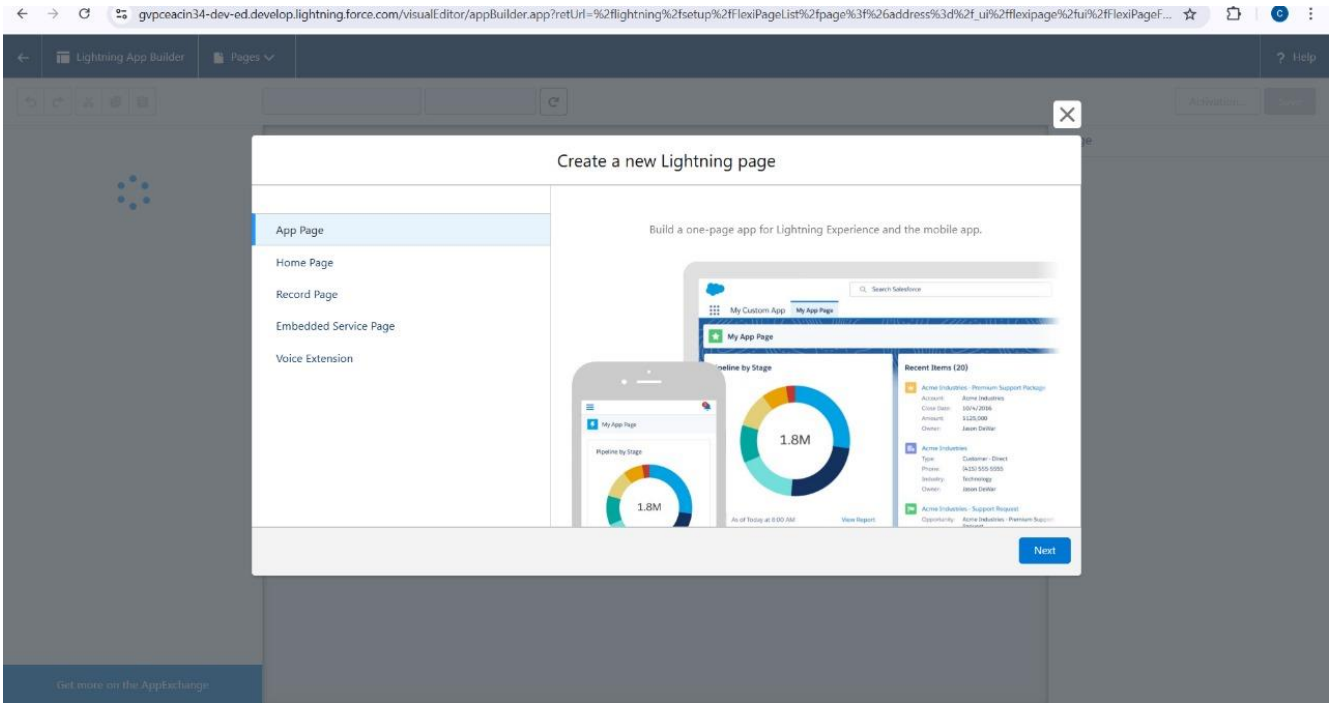
```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

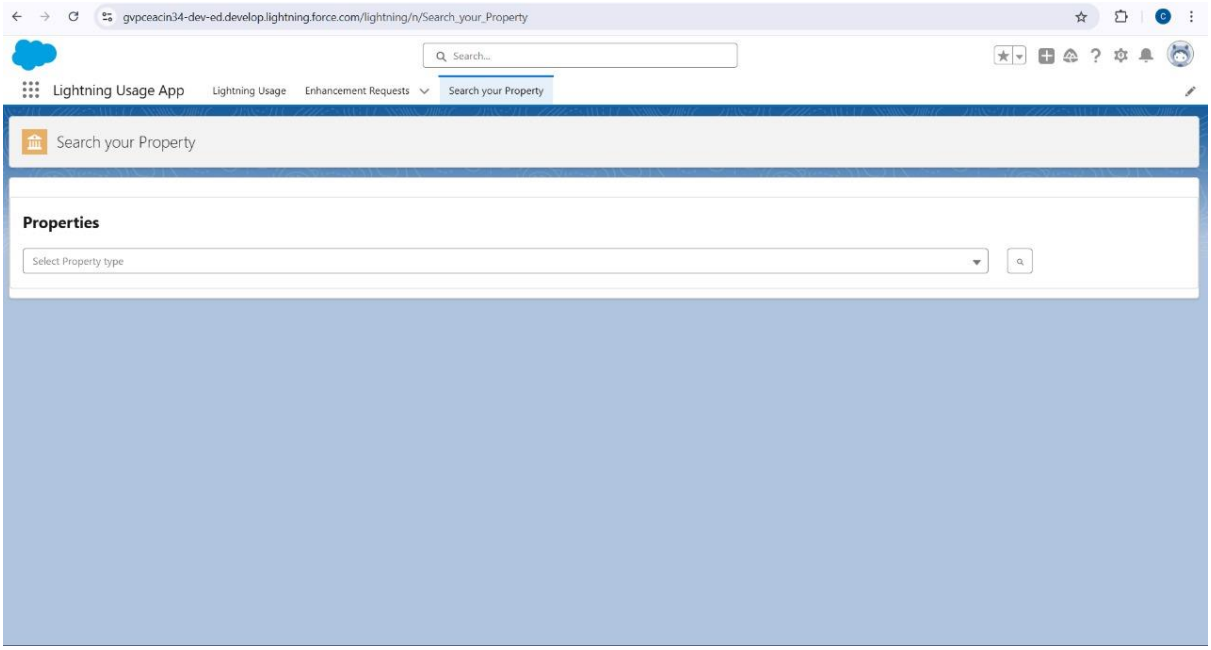
- After saving all the three Codes, Right Click and deploy this component to the org.



Step 13: Drag this Component to Your Apppage

- From Setup >> Go to App Launcher >> Search for Property Details.
- On this Page click on gear icon and click on Edit Page.
- Drag the Component to your App Page and Save the Page.





Step 14: Give access on apex classes to profiles

1. Deploy the LWC component.
2. Drag the component to the App Page.
3. Grant Apex class access to "Manager" and "Customer" profiles.